TP2 - Application de Réalité Augmentée simple

Estimation de pose sans marqueur

1 Introduction

L'objectif de ce second TP est de créer une application de Réalité Augmentée simple faisant apparaître un dragon animé dans un flux vidéo acquis par une webcam (Figure 1). Cette fois, nous n'utiliserons pas un marqueur facilement détectable mais des caractéristiques visuelles automatiquement extraites des images afin d'estimer la pose de la caméra dans le monde réel et ainsi être en mesure de positionner la caméra virtuelle pour générer le rendu.
Comme pour le premier TP, l'application sera programmée en C++ à l'aide des modules de la bibliothèque ViSP [1].

FIGURE 1 – Résultat attendu à l'issu de ce TP

1. https ://visp.inria.fr/

2 Éléments fournis

Vous trouverez dans le dossier "$D:\MV54\visp\_ws$" :

— Les bibliothèques C++ pré-compilés nécessaires pour le développement du programme demandé :
  — ViSP et 2 dépendances (Eigen et OpenCV),
  — CURL pour l'interface avec le Standalone Unity.

Vous trouverez également sur Teams :

— Un Standalone Unity "AR_Engine". La scène de cette application Unity ne contient qu'une caméra virtuelle et le modèle 3D du dragon animé sur un fond purement vert. L'application contient un serveur http afin de rendre la caméra virtuelle contrôlable (changement des paramètres extrinsèques/intrinsèques, acquisition de l'image, etc.) en dehors de l'application via requêtes http.
— Une classe C++ "unityInterface" contenant différentes fonctions permettant de lancer des requêtes http depuis un programme C++ afin de contrôler la caméra virtuelle de l'application Unity.
— Un projet d'exemple "RA_Project" que vous pouvez utiliser comme base de travail. Ce programme vous montre comment :
  — ouvrir et initialiser la webcam usb,
  — acquérir et afficher une frame avec cette webcam,
  — se connecter au serveur http du Standalone Unity,
  — positionner la caméra virtuelle du Standalone Unity,
  — générer et récupérer un rendu de a caméra virtuelle du Standalone Unity.

3 Travail à réaliser

3.1 Détection/description automatique de caractéristiques visuelles

Puisque cette fois, l'application de RA attendue ne repose pas sur l'utilisation d'un marqueur connu facilement détectable, vous devez programmer la détection/description automatique de caractéristiques visuelles dans les images acquises par la webcam (voir la classe vpKeyPoint [2]). Afficher les points d'intérêt détectés, tester différents détecteurs et descripteurs pour différents jeux de paramètres.

3.2 Mise en correspondance

Une solution simple consiste à enregistrer (voir vpImageIo [3]) au préalable une image de référence en vue de dessus de la zone texturée sur lequel vous souhaitez faire apparaître le dragon animé. Puis de mettre en correspondance des points d'intérêt entre cette image de référence et les images acquises par la webcam (toujours

2. https ://visp-doc.inria.fr/doxygen/visp-daily/classvpKeyPoint.html
3. https ://visp-doc.inria.fr/doxygen/visp-daily/classvpImageIo.html

à l'aide la classe vpKeyPoint). Afficher les points d'intérêt matchés (Figure 2), tester différents jeux de paramètres (thresholds, RANSAC, etc.) jusqu'à obtenir des résultats satisfaisants.

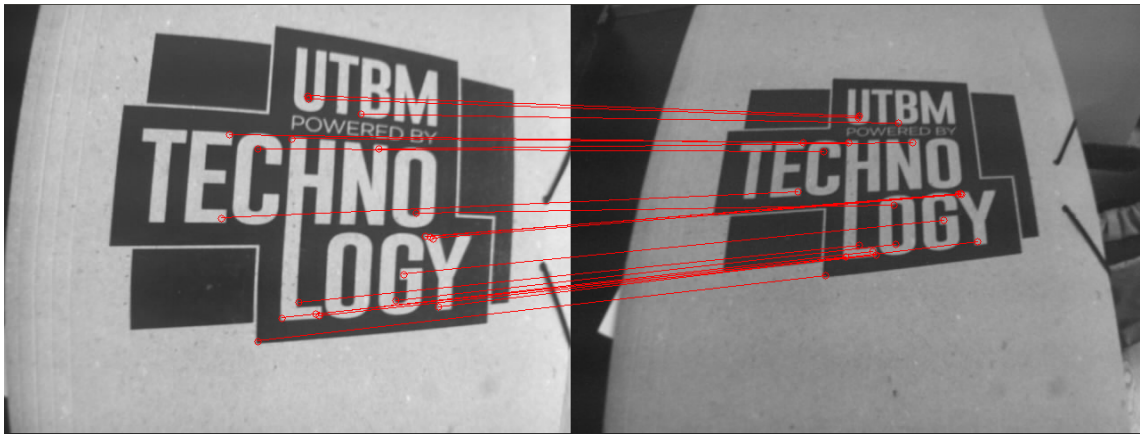

FIGURE 2 – Détection, description et mise en correspondance de points

3.3 Calcul de la pose de la caméra réelle

Il est possible de construire un jeu de points 3D de référence à partir des points d'intérêt issus de l'image de référence. Puis, comme dans le TP précédent, calculer la pose de la caméra réelle par rapport à la scène à partir des correspondances 2D/3D (voir vpPose [4]).

3.4 Génération du rendu virtuel

Lorsque la pose de votre webcam par rapport au marqueur est estimée, il vous est alors possible de modifier les paramètres extrinsèques de la caméra virtuelle du Standalone Unity (voir les fonctions de la classe "unityInterface") afin de positionner et d'orienter cette dernière de la même telle que la pose de la caméra réelle mais vis-à-vis du dragon. Puis, lorsque la caméra virtuelle est en place, vous pouvez alors générer et récupérer un rendu (voir les fonctions de la classe "unityInterface").

3.5 Composition de l'image augmentée

À ce stade, vous avez une image réelle et une image virtuelle qui devraient être "alignées". Il ne vous reste plus qu'à programmer une fonction d'incrustation de l'image virtuelle dans l'image réelle puis d'afficher le résultat. Programmer une méthode d'incrustation plus performante que pour le TP 1 (ex. algorithme de Vlahos, Mishima, Bayesian matting, etc.).

4. https ://visp-doc.inria.fr/doxygen/visp-daily/classvpPose.html

4 Pour aller plus loin...

L'approche implémentée dans ce TP est simple mais possède des limitations importantes. En effet, la qualité de la RA est grandement liée à la qualité de la mise en correspondances. De meilleurs résultats peuvent être obtenus en implémentant un suivi des caractéristiques visuelles [5], en estimant une homographie plutôt qu'une pose "complète" [6] ou encore en changeant de type de caractéristiques visuelles (ex. contours [7], template [8], etc.).

5. https ://visp-doc.inria.fr/doxygen/visp-daily/classvpKltOpencv.html
6. https ://visp-doc.inria.fr/doxygen/visp-daily/classvpHomography.html
7. https ://visp-doc.inria.fr/doxygen/visp-daily/group__module__me.html
8. https ://visp-doc.inria.fr/doxygen/visp-daily/group__module__tt.html